R E M A R K S

Careful review and examination of the subject application are noted and appreciated.

Applicants thank Examiner Ehichioya for the indications of allowable matter for claims 24-28.

INTERVIEW SUMMARY

Applicants' representative, John Ignatowski, spoke with Examiner Ehichioya and his Supervisor A. Kindred on September 1, 2004 via telephone. Applicants' representative argued that the finality of the May 21, 2004 Office Action was premature. The Examiner agreed to withdraw the finality of the May 21, 2004 Office Action and issue a new Office Action. The claims were not discussed. No samples were presented. No agreement was reached regarding the claims.

CLAIM REJECTIONS UNDER 35 U.S.C. §112

The rejection of claims 23, 24, 27 and 29 under 35 U.S.C. §112, first paragraph, best mode, is respectfully traversed and should be withdrawn.

The rejection of claims 23, 24, 27 and 29 under 35 U.S.C. §112, second paragraph, is respectfully traversed and should be withdrawn.

MPEP §2165.03 provides a mandatory two component test for determining if the best mode requirement has been violated. The

Office Action fails to address either component of the test and thus has not meet its burden. As such, the rejection of claims 23, 24, 27 and 29 should be withdrawn.

Regarding claims 23, 24, 27 and 29, the Office Action fails to identify the elements of claims 23, 24, 27 and 29 that allegedly lack antecedent basis. Therefore, the Examiner is respectfully requested to either (i) clearly and concisely identify the claim elements that allegedly lack antecedent basis or (ii) withdraw the rejection.

CLAIM REJECTIONS UNDER 35 U.S.C. §103

The rejection of claims 1, 2, 22 and 30 under 35 U.S.C. §103(a) as being unpatentable over Tang '855 in view of Kucukcakar et al. '229 (hereafter Kucukcakar) is respectfully traversed and should be withdrawn.

The rejection of claims 3-7 and 21 under 35 U.S.C. §103(a) as being unpatentable over Tang in view of Kucukcakar and Kalkstein '903 is respectfully traversed and should be withdrawn.

The rejection of claims 8-10, 23 and 29 under 35 U.S.C. §103(a) as being unpatentable over Tang in view of Kucukcakar and Dabbish et al. '015 (hereafter Dabbish) is respectfully traversed and should be withdrawn.

The rejection of claims 31, 32 and 33 under 35 U.S.C. §103(a) as being unpatentable over Tang, Kucukcakar, Dabbish and Kalkstein is respectfully traversed and should be withdrawn.

Tang concerns method for simultaneous programming of in-system programmable integrated circuits (Title). Kucukcakar concerns a customizable instruction set processor with non-configurable/configurable decoding units and non-configurable/configurable execution units (Title). Kalkstein concerns on-the-fly data re-compression (Title). Dabbish concerns a programmable array logic self-checking system (Title).

Claim 1 provides (in part) a step for generating a programming item from a plurality of parameters that define a program for a programmable logic device. Despite the assertion on page 4 of the Office Action, column 1, line 62 thru column 2, line 30 and column 8, lines 55-58 of Tang appear to be silent regarding the claimed step:

> In the prior art, a "programming command generator" is given a data file which contains only the pattern necessary to program the ISP PLDs on a given system board. The programming command generator derives from the data file all the device dependent parameters, and provides the commands for the programming to occur.

SUMMARY OF THE INVENTION

> In accordance with the present invention, a method for programming, simultaneously, multiple field programmable integrated circuits or devices is provided. According to the present invention, such method includes the steps of: (i) connecting field programmable devices serially in a chain configuration; (ii) constructing a data stream file which contains a data stream for programming the field programmable devices simultaneously, such that the programmable devices are simultaneously scheduled under the data stream (a) to receive instructions, (b) to shift programming data into the field programmable devices, (c) to execute instructions and (d) to shift data out of the field programmable devices; (iii) retrieving from each of a number of program data files, for each of the field programming devices, an individual programming data stream; (iv) filling the data stream file with programming data from each of the individual programming

> data streams thus retrieved; and (v) programming the field programmable devices simultaneously using the data stream file thus composed.
>
> In accordance with a further aspect of the present invention, the individual programming data stream includes programming instructions and programming data. Under this further aspect, programming data are sorted such that rows having a predetermined data pattern (e.g. all `1`s) are placed behind rows not having the predetermined data pattern. In one embodiment, the predetermined data pattern corresponds to a default data pattern of the field programmable device when it is unprogrammed. In accordance with the present invention, programming for such rows can be omitted.
>
> ...
>
> In combination with a 6-bit address generated by programming command generator 404 (FIG. 4), these 132 bits programming bits are shifted into address/data shift register 102 to program a row of AND array 101.

Nowhere in the above text does Tang appear to teach or suggest a step for generating a programming item from a plurality of parameters that define a program for a programmable logic device as presently claimed. The Examiner is respectfully requested to either (i) clearly and concisely identify (a) the alleged plurality of parameters, (b) the alleged programming item and (c) the alleged step of generating the programming item from the parameters or (ii) withdraw the rejection.

Claim 1 further provides a step for compressing a programming item to present a compressed item. Despite the assertion on page 4 of the Office Action, column 8, lines 25-28 of Tang appear to be silent regarding the claimed step:

> Referring back to FIG. 14, upon allocation of ispSTREAM file 1000, step 1403 is completed. Then, the next step, i.e. step 1404 (execution of subroutine "pack_jedec_files" continues), invokes a subroutine "read_jedec_files".

Nowhere in the above text does Tang appear to teach or suggest a step for compressing a programming item to present a compressed

item as presently claimed. Furthermore, the word "pack" in a title of a subroutine does not appear to be evidence that the subroutine performs a compression operation on a programming item. Therefore, *prima facie* obviousness has not been established.

Claim 1 further provides a step for storing a programming item in a programming field of a file in response to generating the programming item. Despite the assertion on page 4 of the Office Action, column 5, lines 13-27 of Tang appear to be silent regarding the claimed step:

> FIG. 1 shows an array map of the ispGAL22V10 device. As shown in FIG. 1, an ispGAL22V10 device is programmed by setting programmable fuses for a 44x132-bit AND array and a 64-bit "user electronic signature" (UES). Programming is accomplished one row at a time by serially shifting a 6-bit row address and 132 bits of program data bits into address/data shift register 102 through the SDI terminal. The 6-bit row address specifies which of the 44 rows of the AND array, or the 64-bit UES, is to be programmed. The 132 bits of programming bits are then provided to implement the desired configuration of the AND array at the specified row or the UES. An 8-bit ID shift register and a 20-bit architectural shift register 104 are provided, respectively, for storing an identification pattern and for specifying configuration information of the ispGAL22V10 device.

Nowhere in the above text does Tang appear to teach or suggest a step for storing a programming item in a programming field of a file in response to generating the programming item as presently claimed.

Furthermore, despite the assertion on page 4 of the Office Action, column 5, lines 50-53 of Kucukcakar appear to be silent regarding a programming field of a file:

> Thus, programmable section 48 is programmed to accept the opcode received at input 40 of instruction execution unit 34 and generate valid control signals for use by datapath 16.

Nowhere in the above text does Kucukcakar appear to teach or suggest a programming field of a file as presently claimed. The Examiner is respectfully requested to either (i) clearly and concisely identify (a) the alleged programming item, (b) the alleged file, (c) the alleged programming field of the file and (d) the alleged step for storing the programming item in the file or (ii) withdraw the rejection.

Claim 1 further provides a step for storing a compressed item in a non-programming field of a file. Despite the assertion on page 4 of the Office Action, column 5, lines 50-53 of Kucukcakar appear to be silent regarding a step for storing in a non-programming field of a file:

> Thus, programmable section 48 is programmed to accept the opcode received at input 40 of instruction execution unit 34 and generate valid control signals for use by datapath 16.

Nowhere in the above text does Kucukcakar appear to teach or suggest a non-programming field of a file as presently claimed. The Examiner is respectfully requested to either (i) clearly and concisely identify (a) the alleged file and (b) the alleged non-programming field of the file or (ii) withdraw the rejection.

Furthermore, the Office Action fails to provide clear and particular evidence of motivation to combine the references. In particular, the Office Action offers no evidence of motivation as required MPEP §2142. Therefore, *prima facie* obviousness has not been established.

Furthermore, the references appear to be non-analogous art. Tang has a primary U.S. classification of 326/38. In

contrast, Kucukcakar has a primary U.S. classification of 712/37. However, no evidence has been provided in the Office Action that Kucukcakar is either (i) within the Applicants' field of endeavor or (ii) reasonably pertinent to the particular problem with which the Applicants' were concerned (MPEP §2141.01(a)). Due to a lack of evidence to the contrary, the U.S. Patent and Trademark Office classifications appear to show that the references are non-analogous art and thus the proposed combination is not obvious. Claim 30 provides language similar to claim 1. As such, the claimed invention is fully patentable over the cited references and the rejection should be withdrawn.

Claim 30 further provides a structure comprising (i) means for generating, (ii) means for compressing, (iii) means for storing a programming item and (iv) means for storing a compressed item. In contrast, both Tang and Kucukcakar appear to be silent regarding the claimed structure. Furthermore, the Office Action provided no evidence or arguments regarding the claimed structure in rejecting claim 30. Therefore, *prima facie* obviousness has not been established. As such, clam 30 is fully patentable over the cited references and the rejection should be withdrawn.

Claim 2 provides a step for storing at least one of a plurality of parameters in a second non-programming field of a file. Despite the assertion on page 5 of the Office Action, column 5, line 67 thru column 6, line 3 of Kucukcakar appear to be silent regarding a second non-programming filed of a file:

> The configured FPGA multiplier receives the control signals and multiplies a data value stored in a register of non-programmable datapath 18 and a data value received on DATA BUS 17.

Nowhere in the above text does Kucukcakar appear to teach or suggest a step for storing at least one of a plurality of parameters in a second non-programming field of a file as presently claimed. The Examiner is respectfully requested to either (i) clearly and concisely identify (a) the alleged file and (b) the alleged second non-programming field of the file or (ii) withdrawn the rejection.

Claims 3-7 and 21 are rejected over Tang, Kucukcakar and Kalkstein. However, the Office Action fails to provide clear and particular evidence of motivation to combine the references. The alleged motivation on page 5 of the Office Action "that the dictionary contains a series of mappings between the original data and the compressed representation of the actual data" is not credited to any of the references or knowledge generally available to one of ordinary skill in the art as required by MPEP §2142. Therefore, *prima facie* obviousness has not been established. The Examiner is respectfully requested to either (i) identify the source of the asserted motivation and, if the source is knowledge generally available to one of ordinary skill in the art, provide evidence thereof or (ii) withdraw the rejection.

Furthermore, the references appear to be non-analogous art. Tang has a primary U.S. classification of 326/38 and Kucukcakar has a primary U.S. classification of 712/37. In

contrast, Kalkstein has a primary U.S. classification of '341/63. However, no evidence has been provided in the Office Action that Kalkstein is either (i) within the Applicants' field of endeavor or (ii) reasonably pertinent to the particular problem with which the Applicants' were concerned (MPEP §2141.01(a)). Due to a lack of evidence to the contrary, the U.S. Patent and Trademark Office classifications appear to show that the references are non-analogous art and thus the proposed combination is not obvious. As such, claims 3-7 and 27 fully patentable over the cited references and the rejection should be withdrawn.

Claim 3 provides a step for generating a dictionary for compressing prior to compressing a programming item. Despite the assertion on page 5 of the Office Action, column 11, lines 47-50 of Kalkstein appear to be silent regarding the claimed step:

> **As each packet is compressed**, a data dictionary is constructed according to the LZ77 algorithm. This data dictionary is composed of items, which can be in one of the following three forms. (Emphasis added)

Nowhere in the above text does Kalkstein appear to teach or suggest a step for generating a dictionary for compressing **prior to compressing** a programming item as presently claimed. Therefore, *prima facie* obviousness has not been established and the rejection should be withdrawn.

Claim 4 provides that the dictionary is generated independently of the step for compressing the programming item. Despite the assertion on page 6 of the Office Action, column 2,

lines 27-29 of Kalkstein appear to be silent regarding an independence between generating a dictionary and compressing:

> In general, data compression techniques encode the original data according to a translation data dictionary referred to herein as the "encoding table".

Nowhere in the above text does Kalkstein appear to teach or suggest that a dictionary is generated independently of a step for compressing a programming item as presently claimed. Therefore, *prima facie* obviousness has not been established and the rejection should be withdrawn.

Claim 7 provides a step for mapping a symbol representation (of an encoded and compressed programming item) to a character representation in response to encoding. Despite the assertion on page 6 of the Office Action, column 2, lines 28-32 of Kalkstein appear to be silent regarding the claimed step:

> An encoding table contains a series of mappings between the original data and the compressed representations of the actual data. For example, the letter "A" may be represented by the binary string "010."

The above text does not appear to teach or suggest the claimed step. In particular, Kalkstein appears to contemplate representing characters by binary strings. However, the claimed step is to represent symbols by characters. Therefore, Tang, Kucukcakar and Kalkstein, alone or in combination, do not appear to teach or suggest a step for mapping a symbol representation to a character representation in response to encoding as presently claimed. As such, claim 7 is fully patentable over the cited references and the rejection should be withdrawn.

Claim 21 provides a step for adding a plurality of delimiters around a compressed item in a non-programming field of a file. Despite the assertion on page 6 of the Office Action, column 11, lines 47-65 of Kalkstein appear to be silent regarding the claimed step:

> As each packet is compressed, a data dictionary is constructed according to the LZ77 algorithm. This data dictionary is composed of items, which can be in one of the following three forms.
>
> The first form, and the most basic building block of the input text, is a "character", typically some 8-bit sequence such as ASCII or EBCDIC representations.
>
> The second form is an "(Offset,Length) pair", in which a subsequent occurrence of a certain character sequence is replaced by a backward reference pointer to some earlier occurrence of that sequence in the text, indicating the location of the earlier occurrence (offset), and the number of characters to be copied (length).
>
> For example, if the text is . . . xxxABCDxABCxx . . . , the second occurrence of ABC can be replaced by the pointer (5,3), indicating that once the string has been processed up to and including Dx, the subsequent characters can be reconstructed by going backwards 5 characters, and copying exactly 3 characters from the data.

Nowhere in the above text does Kalkstein appear to teach or suggest a step for adding a plurality of delimiters around a compressed item in a non-programming field of a file as presently claimed. The Examiner is respectfully requested to either (i) clearly and concisely identify (a) the alleged plurality of delimiters, (b) the alleged compressed item associated with the delimiters, (c) the alleged file and (d) the alleged non-programming field of the file or (ii) withdraw the rejection.

Claims 8-10, 23 and 29 are rejected over Tang, Kucukcakar and Dabbish. However, the Office Action fails to provide clear and particular evidence of motivation to combine the references. The

alleged motivation on page 7 of the Office Action "that error detection code verify each row is programmed is accurately stored" is not credited to any of the references or knowledge generally available to one of ordinary skill in the art as required by MPEP §2142. Therefore, *prima facie* obviousness has not been established. The Examiner is respectfully requested to either (i) identify the source of the asserted motivation and, if the source is knowledge generally available to one of ordinary skill in the art, provide evidence thereof or (ii) withdraw the rejection.

Furthermore, the references appear to be non-analogous art. Tang has a primary U.S. classification of 326/38 and Kucukcakar has a primary U.S. classification of 712/37. In contrast, Dabbish has a primary U.S. classification of 371/22.5. However, no evidence has been provided in the Office Action that Dabbish is either (i) within the Applicants' field of endeavor or (ii) reasonably pertinent to the particular problem with which the Applicants' were concerned (MPEP §2141.01(a)). Due to a lack of evidence to the contrary, the U.S. Patent and Trademark Office classifications appear to show that the references are non-analogous art and thus the proposed combination is not obvious. As such, claims 8-10, 23 and 29 are fully patentable over the cited references and the rejection should be withdrawn.

Claim 8 provides a step for storing an error detection item in a second non-programming field of a file. Despite the assertion on page 7 of the Office Action, column 2, lines 29-39 of

Dabbish appear to be silent regarding a second non-programming field of a file:

> FIG. 1 illustrates a programmable array logic (PAL) device incorporating the PAL self-checking system (100). The external programming equipment (101) programs the PAL (100), via a data bus (102), with polynomial coefficients representing the logic functions of an encryption algorithm and an error detection code generated from the polynomial. The PAL (100) receives the programming data in a tri-state input register (103). the PAL (100) then stores the polynomial coefficients in a programmable array (104) and stores the error detection code in an error detection storage array (105).

Nowhere in the above text does Dabbish appear to teach or suggest a step for storing an error detection item in a second non-programming field **of a file** as presently claimed. The Examiner is respectfully requested to either (i) clearly and concisely identify (a) the alleged file and (b) the alleged second non-programming field of the file or (ii) withdraw the rejection.

Claim 9 provides a step for extracting the error detection item from the file. Despite the assertion on page 7 of the Office Action, the text in column 1, lines 49-57 appear to be silent regarding the claimed step:

> This need is substantially met by the programmable array logic (PAL) self-checking system disclosed herein. The disclosed PAL self-checking system comprises an input register, a programmable array, a fixed array, an error detection register, an error detection circuit and an error detection signal. Polynomial coefficients representing an encryption algorithm and an error detection code generated from the polynomial are stored in the programmable array.

Nowhere in the above text does Dabbish appear to teach or suggest extracting an error detection item from a file as presently claimed. Therefore, *prima facie* obviousness has not been

established. The Examiner is respectfully requested to either (i) clearly and concisely identify the alleged file from which the error detection item is extracted or (ii) withdraw the rejection.

Claim 10 provides that steps (A) through (D) are stored in a storage medium as a computer program that is readable and executable by a computer to generate the file. Despite the assertion on page 7 of the Office Action, the text in column 3, lines 60-63 of Dabbish appear to be silent regarding computer programs:

> The storing and verifying process repeats until all the A(x) coefficients are accurately stored in the programmable NOR array (201).

Nowhere in the above text does Dabbish appear to teach or suggest a computer program stored in a storage medium as presently claimed. Therefore, *prima facie* obviousness has not been established. As such, claim 10 is fully patentable over the cited references and the rejection should be withdrawn.

Claim 23 provides a step for extracting the programming item from the programmable field of the file. Despite the assertion on page 8 of the Office Action, the text in column 2, lines 34-37 of Kucukcakar appear to be silent regarding the claimed step:

> Both non-programmable datapath 18 and programmable datapath 32 receive data from DATA BUS 17, provide functional operations on the data, and transfer data back to DATA BUS 17.

Nowhere in the above text does Kucukcakar appear to teach or suggest extracting a programming item from a programmable field of

a file as presently claimed. Therefore, *prima facie* obviousness has not been established. As such, claim 23 is fully patentable over the cited references and the rejection should be withdrawn.

Claim 29 provides a step for repairing the error detection item in response to a backup programming item failing a validating step. Despite the assertion on page 8 of the Office Action, the text in column 2, lines 45-59 of Dabbish appear to be silent regarding the claimed step:

> After a row of the programmable array (104) is programmed with polynomial coefficients, its storage integrity is verified. The PAL (100) places the tri-state input register (103) in a high impedance state and the tri-state error detection register (107) in an active state. Then the tri-state error detection register (107) reads and stores the contents of the row. Next, the tri-state error detection register (107) and the tri-state input register (103) simultaneously shift their contents into the error detection circuit (106). The error detection circuit (106) compares the two data streams to verify the contents of the row. If the error detection circuit (106) detects an error, it generates an error signal (108). The programming of the error detection storage array (105) is verified similarly.

Nowhere in the above text does Dabbish appear to teach or suggest repairing an error detection item as presently claimed. Therefore, *prima facie* obviousness has not been established. As such, claim 29 is fully patentable over the cited references and the rejection should be withdrawn.

Claim 31 provides a step for extracting the compressed item from the file. Despite the assertion on page 8 of the Office Action, the text in column 4, lines 35-41 of Kalkstein appears to be silent regarding the claimed step:

> (c) selecting an encoding table from a historical array, the historical array including at least one encoding table from

compression of at least one previously compressed data packet, the encoding table having been constructed according to the frequencies of occurrence of a plurality of parsed elements of the at least one previously compressed data packet, independent of data from the packet Pm;

Nowhere in the above text does Kalkstein appear to teach or suggest extracting a compress item from a file as presently claimed. Therefore, *prima facie* obviousness has not been established. As such, claim 31 is fully patentable over the cited references and the rejection should be withdrawn.

Claim 32 provides a step for decompressing a compressed item to present a backup programming item. Despite the assertion on page 9 of the Office Action, the text in column 6, lines 62-67 of Kalkstein appears to be silent regarding the claimed step:

FIG. 3 is a block diagram of a decompressor constructed according to the invention;
FIG. 4 is a flowchart of operations that may be performed by a compression system implemented according to the invention;

Nowhere in the above text does Kalstein appear to teach or suggest a step for decompressing a compressed item to present a backup programming item as presently claimed. Therefore, *prima facie* obviousness has not been established. As such, claim 32 is fully patentable over the cited references and the rejection should be withdrawn.

Claim 33 provides a step for validating the backup programming item with the error detection item. Despite the assertion on page 9 of the Office Action, the text in column 2, lines 58-59 of Dabbish appear to be silent regarding the claimed step:

> The programming of the error detection storage array (105) is verified similarly.

Nowhere in the above text does Dabbish appear to teach or suggest a step for validating a backup programming item with an error detection item as presently claimed. Therefore, *prima facie* obviousness has not been established. As such, claim 33 is fully patentable over the cited references and the rejection should be withdrawn.

Accordingly, the present application is in condition for allowance. Early and favorable action by the Examiner is respectfully solicited.

The Examiner is respectfully invited to call the Applicants' representative should it be deemed beneficial to further advance prosecution of the application.

If any additional fees are due, please charge our office Account No. 50-0541.

Respectfully submitted,

CHRISTOPHER P. MAIORANA, P.C.

Christopher P. Maiorana
Registration No. 42,829
24840 Harper Avenue, Suite 100
St. Clair Shores, MI 48080
(586) 498-0670

Dated: December 10, 2004

Docket No.: 0325.00488